Efficient Approach for Query Optimization in Rough Data

Shruthi Hiremath¹, Pallavi Chandra²

¹, ²School of Computer Science and Engineering, VIT University, Vellore-632014, Tamil Nadu, India

Abstract: In this paper we represent an efficient query optimization technique for the multi-valued rough relational database which follows the indiscernibility relation in its domain. This notion is perceived by using an encoding function to convert a multi-valued attribute to a constant single valued attribute. A simple select-querying technique is provided for selecting the tuples of single-valued attribute from a rough database. We extend the concept of query search to multi-valued attribute. Here we use an encoding function to convert the multi-valued attribute to a single-valued constant attribute to optimize the query search and hence to reduce the response time.

Keywords: rough set theory, rough relational database, indiscernibility relation, encoding function.

1. Introduction

The various applications that are run in the Software and IT industry are mainly uncertain. As a matter of fact any industry that functions today has a lot of uncertain and imprecise data which needs to be compiled. Rough Set Theory [2] was initially introduced by Pawlak to manage uncertain data and analyze the incomplete information effectively. A rough relational database [2] was later implemented to this effect. The major difference between a relational database and an RRDB is that the RRDB can consist of attributes comprising of one or more atomic values unlike relational database where only atomic values of attributes are dealt with. For querying data in a RRDB based on rough set theory it has been explained that for efficient query ,RRDB should be decomposed into standard relational table(semantics of query data is followed) and then use SQL and rough relational operators to get the results. This increases the time and space complexity Further a new concept was introduced where results were based on comparison between equivalence classes rather than values. In this paper we deal with an encoding function [3] which efficiently queries a data by converting a multi-valued attribute into a single-valued attribute thereby reducing the response time for both the lower approximation [2] (certain data) and upper approximation [2] (possible data).

2. Basic Concepts

2.1 Rough Set Theory

Let U be a non-empty set containing the set of all tuples called universal set and R defines an equivalence relation on the universal set U also called the indiscernibility relation. For defining a rough set we define a lower approximation and an upper approximation on a set X where X ⊆ U. Consider an ordered pair attribute A=(U,R).

Lower Approximation:
RX={x ∈ U | [x] R ⊆ U}
This yields certain data.

Upper Approximation:
RX={x ∈ U | [x] R ∩ X≠ ∅}
This yields possible data.

2.2 Rough Relational Database

The rough relational database is similar to the relational database in terms that both comprise of data as a collection of relations containing tuples. These relations are also known as sets and are unordered and non-duplicated.

A relational database is defined as follows S = (U, A, D, R)
For any database U is a set of all tuples, representing the universal set.
A is the attribute set and D is the domain set.

2.2.1 Analysis for relational and rough database

In a classical relational database R is a relation defined over n sets D₁, D₂,..., Dn where Di represents a domain. In a rough relational database R is equivalence classes defined on domain D. Consider aᵢ ∈ A, where dᵢ is a domain defined on aᵢ, rᵢ is an equivalence class on attribute aᵢ. The necessary condition for accessing a tuple t is that t ∈ U where t (aᵢ) should access the value of the attribute aᵢ of that tuple t. This tuple will also be a ⊆ dᵢ.

To define equivalence classes we make use of the indiscernibility relation.

From the table 1 given below the table consists of two attributes ‘Manufacturers’ and ‘Products’.

Using the indiscernibility relation the equivalence classes can be defined as:


Rproduct = [{food, beverages}, {biscuits, confectionery, bakery products}, {dairy products}, {cleaning agents}, {personal care}, {tobacco}]
It is trivial from the ‘manufacturer’ relation that P&G and Proctor and Gamble represent the same company and hence are grouped under the same class.

**Definition 1:** A rough relation is a subset of the set cross product \( P(D_1) \times P(D_2) \times \ldots \times P(D_n) \).

**Definition 2:** An interpretation \( \alpha = (a_1, a_2, \ldots, a_n) \) of a rough tuple \( t = (d_{i1}, d_{i2}, \ldots, d_{in}) \) is any value assignment such that \( a_j \in d_{ij} \) for all \( 1 \leq j \leq n \), \( a_j \) is called a sub-interpretation of \( d_j \).

In Table 1 we present a model of RRDB called ‘FMCG’. Its two attributes are Manufacturers and Products.

**Table 1: Fast Moving Consumer Goods (FMCG) Products**

<table>
<thead>
<tr>
<th>ID</th>
<th>MANUFACTURER</th>
<th>MANUFACTURER BIN</th>
<th>PRODUCTS</th>
<th>PRODUCT BIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>P&amp;G</td>
<td>10000</td>
<td>{Food, Beverages}</td>
<td>100000</td>
</tr>
<tr>
<td>P0</td>
<td>Proctor &amp; Gamble</td>
<td>10000</td>
<td>{cleaning agents, personal care}</td>
<td>000110</td>
</tr>
<tr>
<td>H0</td>
<td>{P&amp;G, Hindustan Unilever Limited}</td>
<td>11000</td>
<td>{Food, Beverages, cleaning agents, personal care}</td>
<td>100110</td>
</tr>
<tr>
<td>H0</td>
<td>HUL</td>
<td>01000</td>
<td>{beverages, food}</td>
<td>100000</td>
</tr>
<tr>
<td>H0</td>
<td>Hindustan Unilever Limited</td>
<td>01000</td>
<td>{cleaning agents, personal care}</td>
<td>000110</td>
</tr>
<tr>
<td>B0</td>
<td>Britannia</td>
<td>00100</td>
<td>{dairy products}</td>
<td>001000</td>
</tr>
<tr>
<td>B0</td>
<td>Britannia Industries</td>
<td>00100</td>
<td>{bakery products}</td>
<td>010000</td>
</tr>
<tr>
<td>B0</td>
<td>Britannia Industries Limited</td>
<td>00100</td>
<td>{biscuits}</td>
<td>010000</td>
</tr>
<tr>
<td>B0</td>
<td>{Britannia, nestle}</td>
<td>00110</td>
<td>{dairy products, biscuits}</td>
<td>011000</td>
</tr>
<tr>
<td>N0</td>
<td>Nestle</td>
<td>00010</td>
<td>{dairy products}</td>
<td>001000</td>
</tr>
<tr>
<td>N0</td>
<td>Nestle S.A.</td>
<td>00010</td>
<td>{bakery products, dairy products}</td>
<td>011000</td>
</tr>
<tr>
<td>I00</td>
<td>ITC</td>
<td>00001</td>
<td>{Tobacco}</td>
<td>000001</td>
</tr>
<tr>
<td>I00</td>
<td>ITC Limited</td>
<td>00001</td>
<td>{Foods, Confectionery}</td>
<td>110000</td>
</tr>
<tr>
<td>I00</td>
<td>Indian Tobacco Limited</td>
<td>00001</td>
<td>{personal care}</td>
<td>000010</td>
</tr>
<tr>
<td>I00</td>
<td>{ITC, HUL}</td>
<td>01001</td>
<td>{Food, personal care}</td>
<td>100010</td>
</tr>
</tbody>
</table>

Here, in this real-time application we have taken two attributes MANUFACTURER and PRODUCTS. After applying encoding function the attributes are extended to two more attributes namely MANUFACTURER_BIN and PRODUCT_BIN. The domain \( D \) and relation \( R \) for the table can be defined as:

- \( D_{\text{manufacturer}} = \{ \text{P&G, Proctor & Gamble, Hindustan Unilever Limited, HUL, Nestle, Nestle S.A., ITC, Indian Tobacco Limited, ITC Limited, Britannia, Britannia Industries, Britannia Industries Limited} \} \)
- \( R_{\text{manufacturer}} = \{ \{ \text{P & G, Proctor & Gamble} \}, \{ \text{Hindustan Unilever Limited, HUL} \}, \{ \text{Britannia, Britannia Industries, Britannia Industries Limited} \}, \{ \text{Nestle, Nestle S.A.} \}, \{ \text{ITC, Indian Tobacco Limited, ITC Limited} \} \} \)

**Dproducts** = \{ Food, Beverages, Cleaning agents, Personal care, Dairy products, Bakery Products, Biscuits, Tobacco, Confectionery \}

**Rproducts** = \{ [Food, Beverages], {Biscuits, Confectionery, Bakery products}, {Dairy products}, {Cleaning agent}, {Personal care}, {Tobacco} \}

### 3. The Encoding Function

To reduce the query response time we hereby define an encoding function for optimization:

1. Calculate the number of equivalent classes defined for each attribute \( a_i \) defined over a domain \( D(a_i) \).
2. Assign that many number of bits as calculated in (1).
3. For any given value of a tuple \( t \) for an attribute \( a_j \), check in which equivalent class the value is present. Assign 1 to the position corresponding to the class if present else 0.
4. Repeat the above steps and compute the encoded values for all the attributes in \( D \).
5. If a multi-valued attribute is present then the encoding function for the same is defined using values of different equivalent classes defined on domain \( D \). It will be computed by the OR operation of the individual encoded values calculated using (3).

**Algorithm 1**

**4. Algorithm 1**

Suppose the origin select-condition is \( "a = v" \), \( a \) is an attribute and its domain is \( D(a) \). The encoding function for \( a \) is \( \text{ENCODE}(a, v) \).

1. Calculate the value of \( \text{ENCODE}(a, v) \) and note the result as \( c \), that is \( c = \text{ENCODE}(a, v) \).
2. The search condition of certain data querying can be modified to “a_BIN = c”.
3. The search condition of possible data querying can be modified to “a_BIN >= c ∧ a_BIN & c = c”, “a_BIN >= c” is an additional condition, and it can narrow the scope of search. According to the Algorithm 1, we can get the more common expression of our new method.

### 4.2. Algorithm 2

Suppose the origin select-condition is “a1 = v1 ∧ a2 = v2 … ∧ an = vn”, for all 1≤i≤n, ai is an attribute and its domain is Da, and ai_BIN is the encoding filed of ai; vi is an arbitrary value and vi ⊆ Da.

1. For all 1≤i≤n, calculate the value of ENCODE (ai, vi) and note the result as ci.
2. The search condition of certain data querying can be modified to “a1_BIN = c1 ∧ a2_BIN = c2 … ∧ an_BIN = cn”.
3. The search condition of possible data querying can be modified to “(a1_BIN >= c1 ∧ a1_BIN & c1 = c1) ∧ (a2_BIN >= c2 ∧ a2_BIN & c2 = c2) … ∧ (an_BIN >= cn ∧ an_BIN & cn = cn)”.  

#### 5. Experiment

To get the results for certain data we use the query format a_BIN = c. This is the lower approximation. Refer Figure 1.

To get the results of for possible data we use the query format a_BIN >= c ∧ a_BIN & c = c. This is the upper approximation. Refer Figure 2.

Consider the query: “select ID from FMCG where MANUFACTURER_BIN >= 10000 and MANUFACTURER_BIN & 10000 = 10000”.

The result obtained after executing MANUFACTURER_BIN >= 10000 is ID = {P001, P002, H001}. Now ANDing these values with 10000 we get possible data i.e. ID = {P001, P002, H001}.

#### Table 2: Certain Data Querying

<table>
<thead>
<tr>
<th>Certain Data Querying</th>
<th>Total Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Encode Algorithm</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

#### Table 3: Possible Data Querying

<table>
<thead>
<tr>
<th>Possible Data Querying</th>
<th>Total Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Encode Algorithm</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

### 6. Conclusion

In this paper we present a solution to reduce response time for querying multi-valued attribute in rough relational database. We have presented this by encoding multi-valued into single-valued attribute. We have also defined as sample query which makes use of the encoded values obtained after applying the encoding function on the attributes in the table.

### 7. Future Work

Further research could be carried out when an issue such as a conflict that arises when a query tries to access a tuple of any attribute ai defined on a domain di which contains elements belonging to the same equivalent class, the problem that arises when we try to access is that it will not be able to distinguish which element it should retrieve. This problem may be solved by assigning separate bits for the equivalence classes.

### References


Author Profile

Shruthi Hiremath is a third year undergrad student currently pursuing her Bachelor’s degree in Computer Science and Engineering from Vellore Institute of Technology, Vellore. She has an aptitude for research work and is currently working on a project concerning provision of health-care solutions to the employees of the municipal corporation of the Pune city, Maharashtra, India. She aspires to do a Master’s degree in Computer Engineering, her subjects of interest being Database Systems, Rough Sets, Software Engineering, Soft Computing and Computer Networks.

Pallavi Chandra is a third year undergrad student also currently pursuing her Bachelor’s degree in Computer Science and Engineering from Vellore Institute of Technology, Vellore. Her areas of interest lie in Cryptography, Database Systems and Rough Sets. She is presently working on a project with BSNL. She wishes to appear for the Indian Administration Examinations.